Modern Engineering

January 9 | Day 2

Prudential Financial

GENERAL ASSEMBLY

Whilst we wait....

In chat write:

Your name - role and a recent impulse buy.

 © 2024 General Assembly

Remember Your To Do List

- Be engaged and ask questions
  Start every session with your microphone **muted** and **camera on**
- Communicate with us
- Don’t be afraid of breaking things
- Don’t worry about writing things down
- Try not to get distracted by your inbox or phone calls
- Help your peers out

You will receive digital copies of the slides in your inbox after today.

 © 2024 General Assembly

LESSON ROADMAP

Postman

Applying Postman to our API

Discover SQL

Creating and Modifying Queries

Stakeholder Questions to Queries

 © 2023 General Assembly

MEF MODULE 1 DAY 2: Postman and Intro to SQL

Schedule	
9:00–9:15 am	Welcome and Warm-Up
9:15–10:30 am	**Postman Walk Thru**
10:30–12:30 pm	**Using Postman to make DMV Queries**
12:30–1:30 pm	Lunch
1:30–2:00 pm	**The Structure of SQL**
2:00 pm–4:50 pm	**Querying Databases**
4:50–5:00 pm	Bring It Home

LEARNING OBJECTIVES

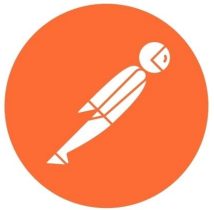
1. Understand the role of Postman in automating API development testing
2. Use Postman to send API requests and parse responses
3. Compare between database structure options and make recommendations based on use case
4. Construct SQL queries to make sophisticated requests from a database

Modern Engineering

Postman Walk-through

Guided Walk-Through:
What is Postman?

Postman is an API platform for building and using APIs.



What is Postman?

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

API Tools

A comprehensive set of tools that help accelerate the API Lifecycle - from design, testing, documentation, and mocking to discovery.

API Repository

Easily store, iterate and collaborate around all your API artifacts on one central platform used across teams.

Workspaces

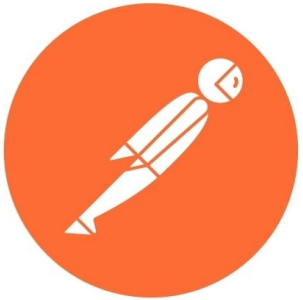
Organize your API work and collaborate with teammates across your organization or stakeholders across the world.

Governance

Improve the quality of APIs with governance rules that ensure APIs are designed, built, tested, and distributed meeting organizational standards.

Guided Walk-Through:

Planning our API's: Review Vehicle API

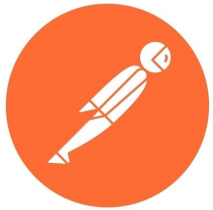


1. Start the [simplified DMV API application]
   a. We'll follow directions provided in the readme.md file

2. Open Postman

3. Let's use Postman to explore our API:
   a. Driver
   b. Plate
   c. Makes
   d. Models

Guided Walk-Through:
Setup Postman - Open The App



Open a terminal

Enter `/opt/Postman/app/Postman`

Postman should load

Guided Walk-Through:

Using Postman - Our First Request

Enter the url to request here

localhost:3001/api/drivers

Save

GET

localhost:3001/api/drivers

Send

 © 2024 General Assembly

Guided Walk-Through:
Using Postman - Our First Response

The response should be displayed like so:

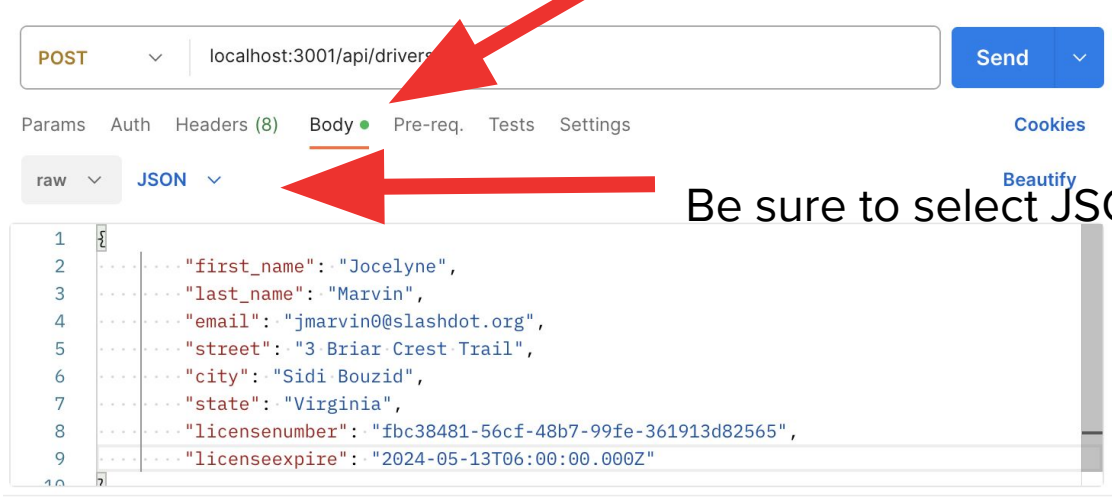
Body ▾ 200 OK 173 ms 248.8 KB Save Response ▾

Pretty Raw Preview Visualize JSON ▾


```
[
    {
        "id": 1,
        "first_name": "Jocelyne",
        "last_name": "Marvin",
        "email": "jmarvin0@slashdot.org",
        "street": "3 Briar Crest Trail",
        "city": "Sidi Bouzid",
        "state": "Virginia",
        "licensenumber": "fbc38481-56cf-48b7-99fe-361913d82565",
        "licenseexpire": "2024-05-13T06:00:00.000Z"
    },
```

Guided Walk-Through:
Using Postman - Sending Request Bodies

Select the Body tab here

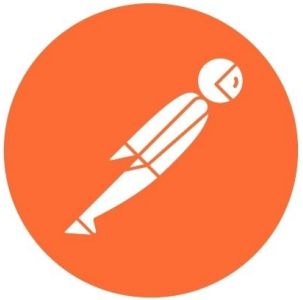


Be sure to select JSON when sending data

Guided Walk-Through:

Explore Vehicle Service with Postman

1. What is the Make_ID for Toyota?
2. Get all Models for Jaguar
3. What is the Model_ID for a Ford Flex?

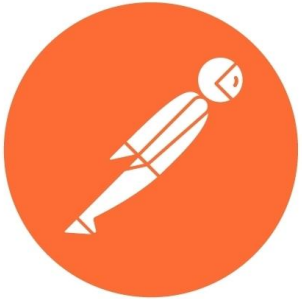


 © 2024 General Assembly

Partner Exercise:

Create And Update Resources

30 minutes



1. Create/Add a Driver
2. Get all Drivers
3. Get the Driver by its ID
4. Update the Driver
5. Delete the Driver

When you’re done, try testing every route of every resource

Time for Lunch!

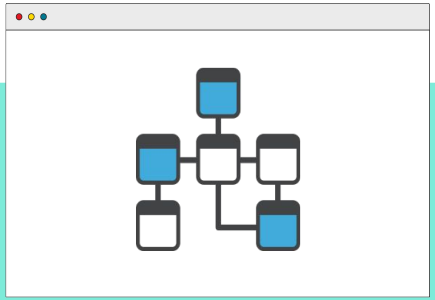
See you in an hour!

 © 2024 General Assembly

Prudential - Modern Engineering

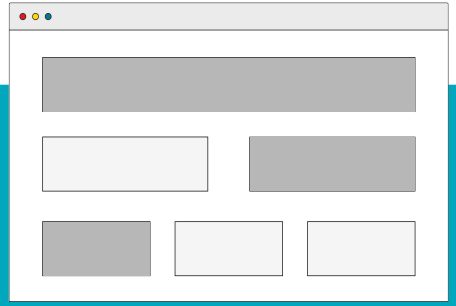
Database Structures: SQL v. NoSQL

Database Types



Relational

Relational databases represent and **store data in tables and rows.** Examples include Postgres, MySQL and Oracle.



Non-Relational

A non-relational database is any database that **does not use the tabular schema** of rows and columns like in relational databases.

Non-Relational Databases (NoSQL)

NoSQL databases provide a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. It's basically any type of database that doesn't use SQL.
A few popular examples include:

 © 2024 General Assembly

NoSQL Databases

How data is stored in MongoDB should look a little familiar:

```
    {
      "data": "car",
      "make": "ford",
      "model": "focus"
      "details":
    {
      "color" : "blue",
      "mileage" : "54019"
    }
}
```

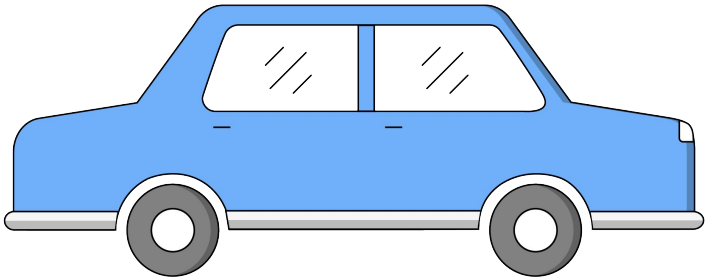


Image 3: Code Block of JSON data stored in MongoDB

One Datastore for Each Microservice

Each microservice has a discrete datastore. You want the team for each microservice to **choose the database that best suits the service.**

Remember: with data spanning multiple microservices, we don't have access to traditional transactions to ensure data consistency.

JOINS

SQL *Query*

```
SELECT * FROM players P
INNER JOIN Teams T ON P.team
= P.team
WHERE P.yards = 50
AND P.score = true
ORDER BY player.number DESC
```

Image 1: SQL Query Code Block using JOINS clause

Mongo *Pipeline*

```
db.teams.aggregate([
  { $match : { name : 'NorthBend' } },
  { $unwind : $players },
  { $game : {
      _id : 0,
      player.yards : 50,
      player.score : true
      } },
  { $sort : {
       player.number' : -1
      } }
]).pretty()
```

Image 2: Mongo Pipeline Code Block for Non-Relational joins using .aggregate pipeline

Choosing Your Database

- Fully Relational
- Transaction Support
- Easy to optimize and index

- Doesn't require a server
- Simple
- Limited write performance
- Limited optimizations

- Non-relational
- Easy (and massive) horizontal scaling
- Limited indexing
- Cross-references slow

Relational vs. Non-Relational: Scaling

Relational:

- Shards
- Read replicas
- Vertical Scaling

NoSQL:

- Shards (Easily)
- Read replicas (less an issue)
- Vertical Scaling (no need)

The tree-like structure is highly conducive to shards and horizontal scaling. Most no-sql systems have larger maximal sizes.

Larger Scales

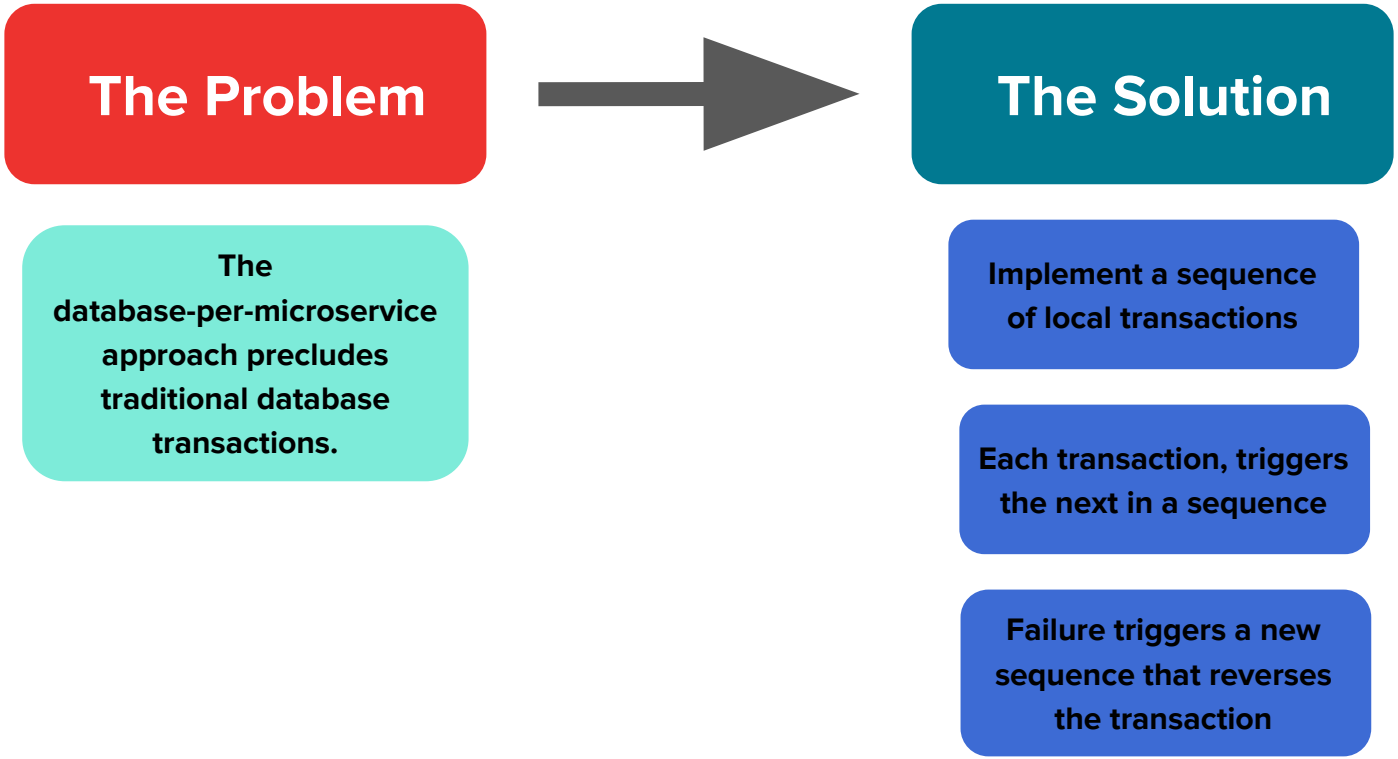
- At data-lake scales, data generally becomes less relational
- Queries often can be measured in minutes to run rather than milliseconds
- Queries have measurable cost impact as scales increase.

Examples:

Amazon Redshift

 © 2024 General Assembly

Saga



 © 2024 General Assembly

Saga

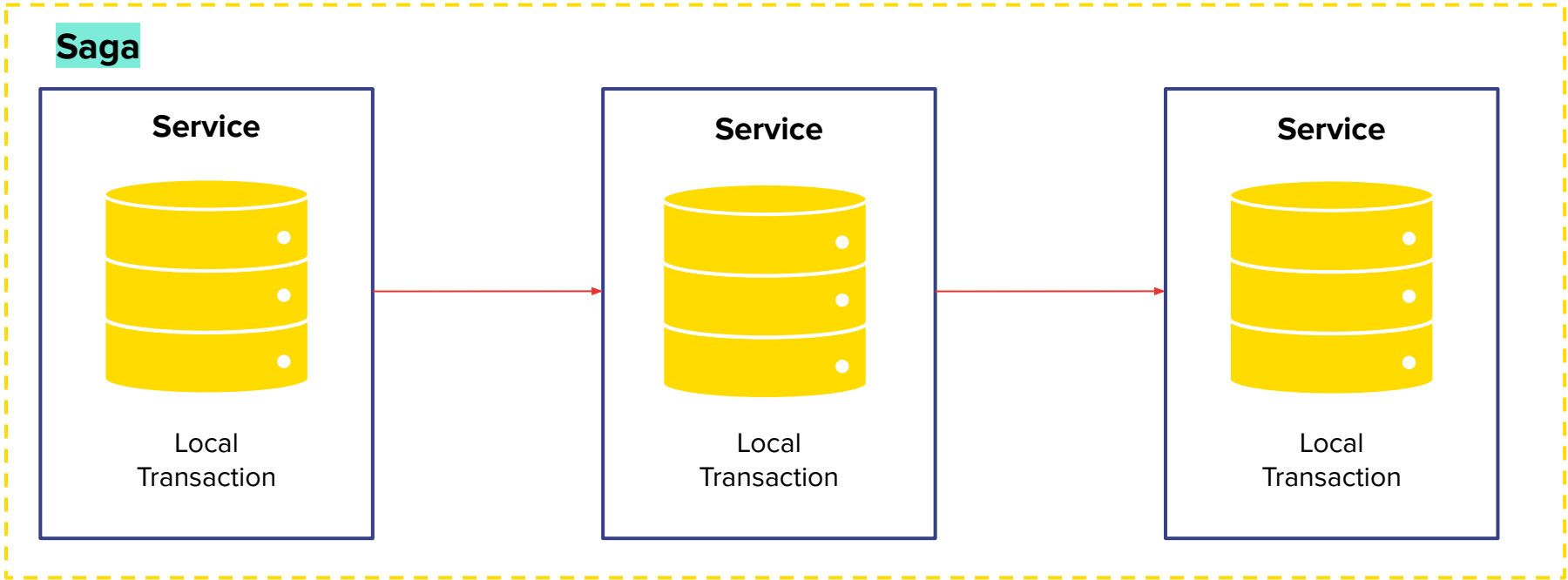


Image 3: Saga Diagram (High Level Overview)

Saga: Key Considerations

- **Significantly more complex** to build and debug
- True data rollbacks are not possible as data is committed to individual databases
- This can be combined with the publisher-subscriber pattern to coordinate the transactions
    - Each local transaction publishes both success and fail events
    - Each local transaction listens for triggering events

 © 2024 General Assembly

Saga

When to Use

- No single point of failure
- Rough equivalence to transactions in data

Don’t Use

- Workflows can be very confusing
- Risk of cyclic dependency when services trigger events on each other
- Transactions across services are tightly coupled

© 2024 General Assembly

Recap

- Database schemes
- Types of databases
- Saga pattern
- Data domain considerations when breaking up legacy applications

Any Questions

Prudential - Modern Engineering

Discover SQL

What's Coming

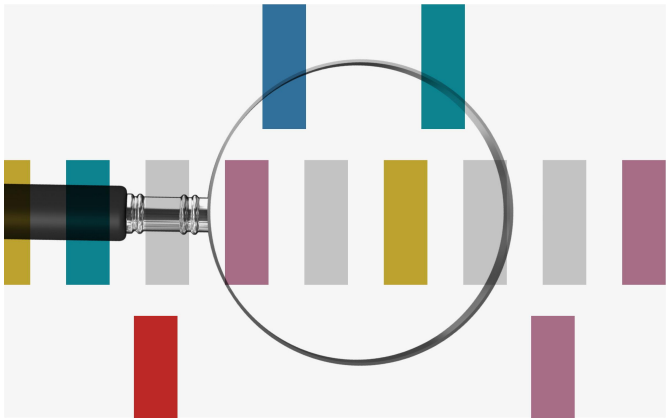
- Describe a relational database.
- Illustrate the structure of a database using an entity relationship diagram.

What Is SQL?

SQL is short for **Structured Query Language**, pronounced *“see-quel.”*

It’s a language you can use to *query* information from your data. With SQL, you can ask questions and make requests such as:

- “How many users logged in this week?”
- “Show me the posts by this user from the past month.”



 © 2024 General Assembly

SQL

- Can query, retrieve, and aggregate **millions** of records.
- Cloud-based data query and retrieval is not limited to your local computer system.
- Can organize data tables.
- Allows users to remotely interact with large data sets in production environments.

Excel

- Has a fixed upper **limit** of **1,048,576 rows** and **16,384 columns**.
- Is limited by your computer's available memory and system resources.
- Can analyze and visualize small amounts of data.

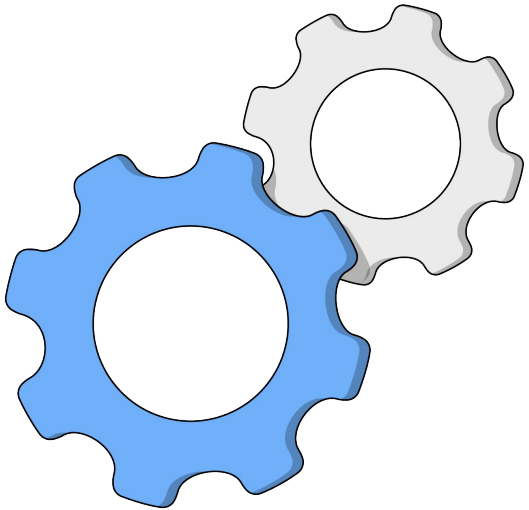
Benefits of SQL

- Simple to use.
- Industry standard for nearly all relational databases.
- Approachable syntax.
- Supportive community.
- Available in free and open versions.

Limitations of SQL

Although SQL has quite a few impressive superpowers, there are also limitations:

- SQL is **not** a data visualization tool.
- SQL is **not** typically used to analyze data.
- SQL is **not** a complete replacement for Excel, given its lack of visualization functionality.
    - It's normally used **in conjunction with Excel, Tableau, and other tools**.



 © 2024 General Assembly

The Structure of SQL

Navigating a SQL Database

PgAdmin PostgreSQL

- PostgreSQL is an open-source SQL standard compliant RDBMS (relational database management system).
- PgAdmin is a popular and feature-rich open-source administration and development platform for PostgreSQL.
- PostgreSQL is extensible and has strong online community support.

 © 2024 General Assembly

Guided Walk-Through:
Navigating the Superstore Data Set

We'll be working with a single data set throughout the day to apply what we're learning. Let's go through the following steps to get started:

1. Connect to the SQL database that we'll be using for this unit.
   a. Click this link to access your nearest host browser.

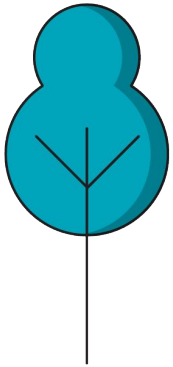
Username: analytics_student@generalassemb.ly
Password: analyticsga

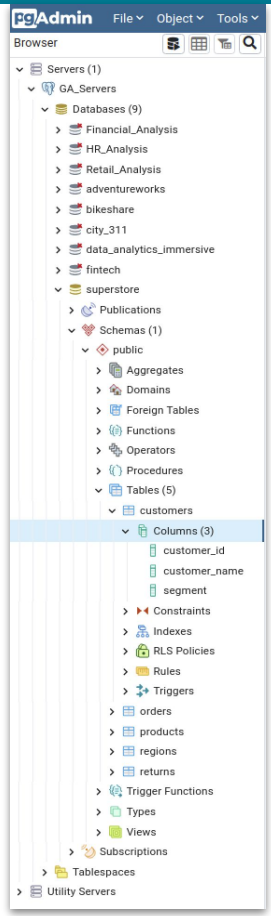
2. Explore the functions of the client software (execute, stop, save, new query).
3. Look at the first and last 100 rows of the data from the tables using the menus.
4. Review how the column properties are defined using the menus.

 © 2024 General Assembly

Guided Walk-Through:
Superstore Data Directory Tree

- Servers (*pgAdmin can be configured for multiple server connections*).
  - Databases ➔ Superstore (our database for today).
    - Schemas ➔ Public (our collection of tables).
      - Tables (this would be the Superstore table names).
        - Columns (for each table).





Telling the Story of One Row

To understand our data, we need to know what's stored. One way to do this is by *telling the story of one row*:

- Read across one row of your data to really see and understand the values contained in every column.

This will help you get to know what's in your data set or propel you to investigate further if you don't understand a particular column.

 © 2024 General Assembly

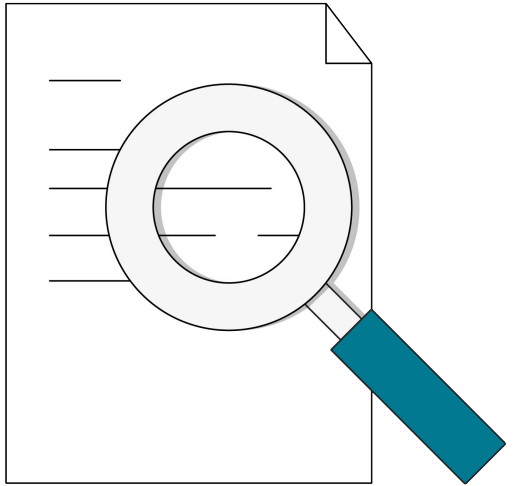
PostgreSQL Data Types

	Explanation
character	Limited-length text-based string
text	Unlimited amount of text
numeric	Floating-point number
integer	Whole number
timestamp	Date and time
date	Just the date

Previewing Your Data Set - Data Analysis Workflow

Steps in a Data Analysis Workflow

1. Defining the Question
2. Data Collection
3. Data Cleaning and Preparation
4. Data Analysis
5. Interpretation and Reporting



 © 2024 General Assembly

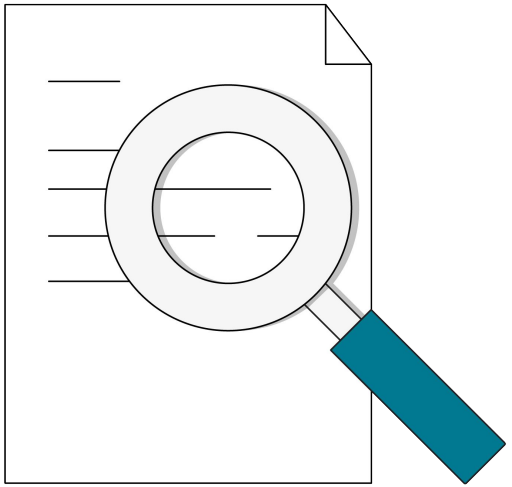
Discussion:

Previewing Your Data Set

Examining 100 rows of data may help us better understand if this is the data set we need to answer our research questions. This subset allows us to preview the data.

How does “previewing” the data connect to the Data Analytics Workflow?

Is this useful? Why or why not?



Guided Walk-Through:
Viewing the Tables

To view the contents of a table, let's navigate to the tables in the Superstore database's public schema.

- There, you should see five tables: Customers, Orders, Products, Regions, and Returns.
- To view the data, right click on the table, go to "View Data," and select the top 100 rows.

Revealing contents of the tables and keeping the column names in view will assist you as you author queries.

Group Exercise:

Transactional vs. Reference Tables

10 Minutes

Transactional tables are large and updated frequently, whereas **reference** tables are rarely modified.

Now that we've seen the five tables — Customers, Orders, Products, Regions, and Returns — which ones do you think are transactional and which are reference? In your breakout room discuss.

 © 2024 General Assembly

Partner Exercise:

Describing Stored Data

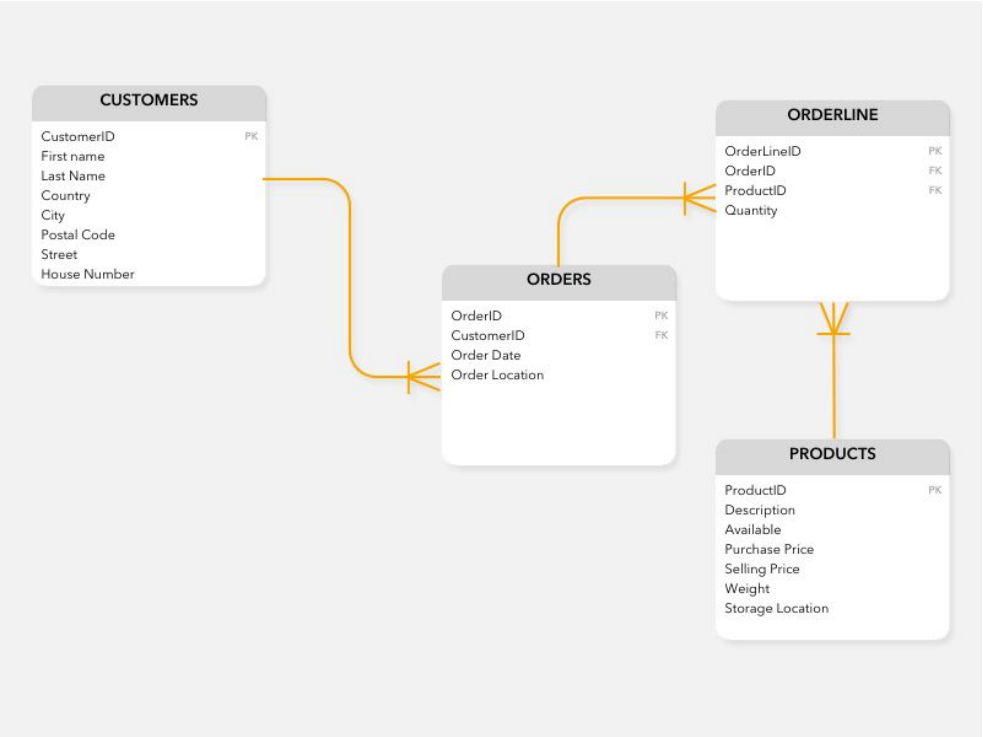
20 minutes

Let's take a closer look at the tables in the Superstore data set and explore the data they contain.

- Characterize each table as either transactional or reference.
- Take five minutes and message with your partner about the database:
  - Write a few sentences describing the data stored in each table.
  - Note the data types assigned to each column.
  - Which columns could serve as links between tables later in our data exploration?

 © 2024 General Assembly

Entity Relationship Diagram (ERD)



An **entity** is a **component of data.**

An **entity relationship diagram** (ERD) shows the relationships of **entity sets** stored in a database.

ERDs help us illustrate the **logical structure** of databases.

PK: Primary key
FK: Foreign key, all unique identifiers

Adapted from: Guru99.com

Solo Exercise:

Over to You

10 minutes

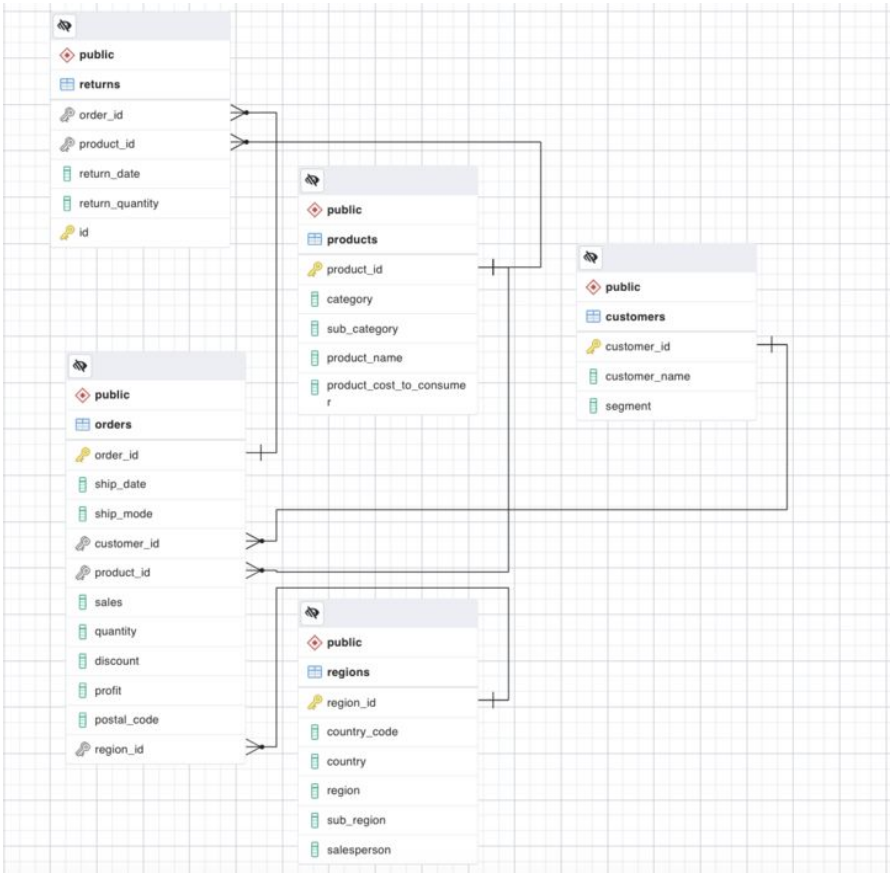
Now that you've seen a sample ERD, use this time to create one for our Superstore database on paper or in a program of your choice.

STORES	COUNTIES	SALES	PRODUCTS
store	county	date	item_no
name	population	category_name	category_name
store_address		description	item_description
store_status		item	proof
		county	case_cost
		store	pack
		total	

 © 2024 General Assembly

Solo Exercise:

Over to You | Solution



Discover SQL

Querying Databases

The Structure of SQL

Querying Databases

Filtering and Aggregating Data

JOINing Tables

What's Next

- Write queries to retrieve data from a SQL database.

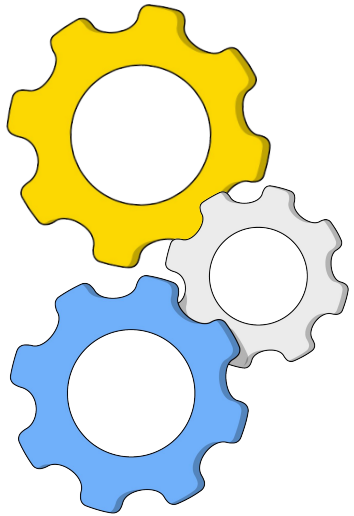
The Two Main Clauses of a SQL Query

SELECT

- Allows you to select certain *columns* from a table.
- Determines which *columns* of information are downloaded.

FROM

- Specifies the *tables* from which the query extracts data.



 © 2024 General Assembly

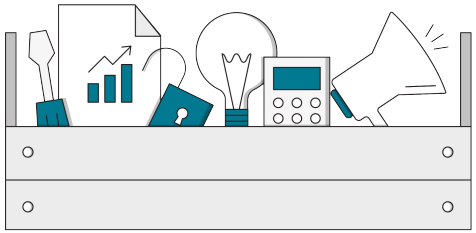
Did You Know...?

You ran a query already! When we select the top 100 rows of a table using the “View Data” menu or “Query Tool” menu selections, this SQL statement runs in the background:

SELECT * FROM products LIMIT 100;

Take a closer look at the syntax above and try answering these questions:

- ❏ What does * mean?
- ❏ What does **FROM products** mean?
- ❏ What does the **LIMIT** do?



The SQL Query Order of Construction

SELECT picks the columns.

FROM points to the table.

WHERE puts filters on rows.

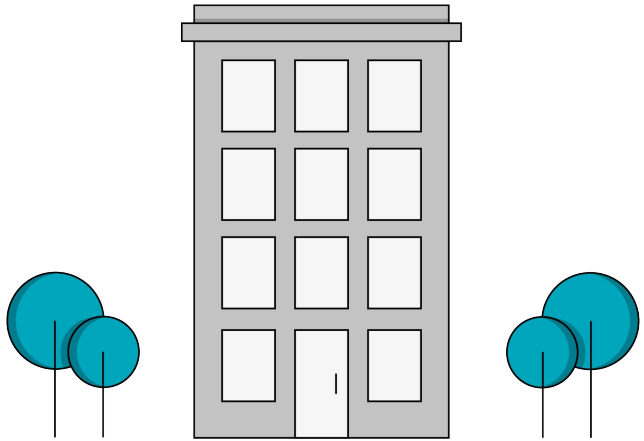
GROUP BY aggregates across values of a variable.

HAVING filters aggregated values ***after*** they have been grouped.

ORDER BY sorts the results.

LIMIT limits results to the first **n** rows.

The Query Building



Practicing Good SQL Grammar

Common punctuation:

- Signal the end of your SQL query with a semicolon (;).
- Commas separate column names in an output list (,).
- Use single quotations around text/strings ('Nokia').

```
SELECT column1, column2
FROM table -- important note
WHERE column1 = 'Some Text';
```

Query code spacing:

- SQL only requires a single white space to separate elements.
- Carriage returns are often used to enhance readability.

Notes within queries:

- Always provide comments (source, revision, author, etc.).
- Comments are made after typing double dashes or the pair /* */.

 © 2024 General Assembly

Guided Walk-Through:
Modifying a Query

All queries can be run in the pgAdmin SQL window. For the remainder of the lesson, we'll practice modifying queries.

If we wanted to return the product ID and product name, which table would we use?

First, we can tell **SELECT** which columns or variables we want:

```
SELECT product_id, product_name
FROM products;
```

 © 2024 General Assembly

Group Exercise:
Modifying a Query

5 minutes

Work with your group to return all countries, regions, and salespeople. Which table would you use and which columns do you need to return?

 © 2024 General Assembly

Group Exercise:
Modifying a Query | Solution

Your query should look like this:

```
SELECT country, region, salesperson
FROM regions;
```

Introducing SELECT DISTINCT

SELECT DISTINCT returns a unique combination of values for all columns selected.

Every row is therefore a unique combination of values and results in a de-duplicated table.

```
SELECT DISTINCT column1, column2
FROM table -- important note
WHERE column1 = 'Some Text';
```

 © 2024 General Assembly

Guided Walk-Through:

SELECT DISTINCT

We can add **DISTINCT** to the query statement to eliminate duplicates **rows** of categories and subcategories:

```
SELECT DISTINCT category, sub_category
FROM products;
```

 © 2024 General Assembly

The SQL Query Order of Construction

SELECT picks the columns.

FROM points to the table.

WHERE puts filters on rows.

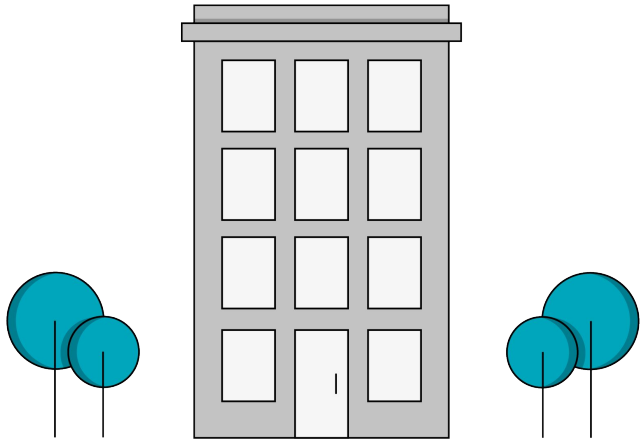
GROUP BY aggregates across values of a variable.

HAVING filters aggregated values ***after*** they have been grouped.

ORDER BY sorts the results.

LIMIT limits results to the first **n** rows.

The Query Building



Guided Walk-Through:

Sorting Results With **ORDER BY**

ORDER BY sorts results in ascending or descending order. After **ORDER BY** is a number that indicates the column by which you're sorting.

The default sort order is ascending, but you can specify ascending (**ASC**) or descending (**DESC**) to determine the sort order.

```
SELECT *

FROM products

ORDER BY 1 ASC;
```

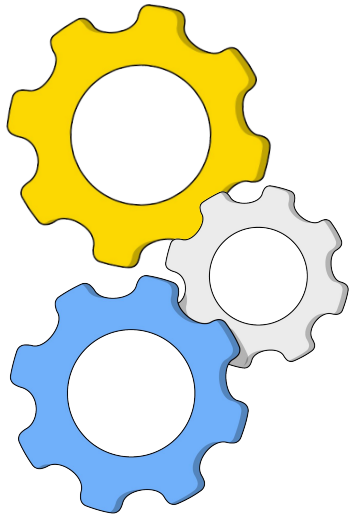
Querying Databases

WHERE Conditions

Introducing the WHERE Clause

WHERE

Allows you to select certain *rows* from a table based on a single or multiple conditions.



 © 2024 General Assembly

The SQL Query Order of Construction

SELECT picks the columns.

FROM points to the table.

WHERE puts filters on rows.

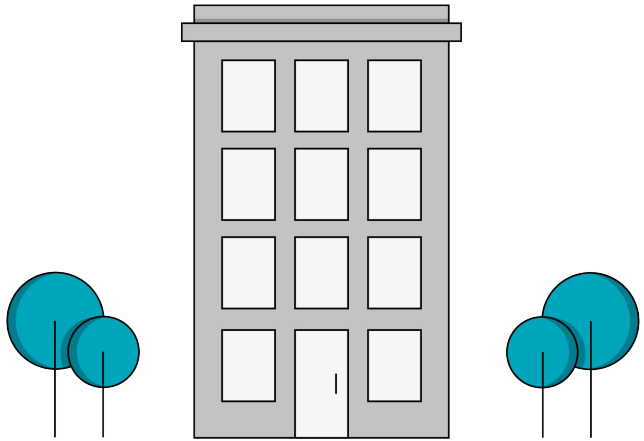
GROUP BY aggregates across values of a variable.

HAVING filters aggregated values ***after*** they have been grouped.

ORDER BY sorts the results.

LIMIT limits results to the first **n** rows.

The Query Building



Guided Walk-Through:
Filtering With **WHERE**

The **WHERE** clause filters rows by setting a criteria:

```
SELECT *
FROM products
WHERE sub_category = 'Furnishings';
```

 © 2024 General Assembly

A Little Filtering Goes a Long Way

SQL has a variety of commands and operators to ask filtering questions. To make the most out of filtering with WHERE...

- Carefully consider the data collection process for **potential irregularities** in your data.
- Select your command tools to make sure no data is left behind.
- **Validate your results** by including the testing columns in your output.
- Consider how to **reconcile your results** with the rest of the table to ensure accurate insights.

Guided Walk-Through:

Basic Operators for WHERE

<>, != Not equal to.

>, >= Greater than; greater than or equal to.

<, <= Less than; less than or equal to.

Which countries are not in the “EMEA” region?

```
SELECT DISTINCT country
FROM regions
WHERE region != 'EMEA';
```

 © 2024 General Assembly

Guided Walk-Through:
Basic Operators for WHERE (Cont.)

Which orders have more than $90 in profit?

```
SELECT *
FROM orders
WHERE profit > 90;
```

Guided Walk-Through:
Basic Operators for WHERE (Cont.)

Which returns include more than one quantity of an item?

```
SELECT *
FROM returns
WHERE return_quantity > 1;
```

 © 2024 General Assembly

Guided Walk-Through:
Filtering With AND, OR, and ()

Some additional query methods to try in the WHERE clause:

- **AND**: Returns if both conditions are true.
- **OR**: Returns if either condition is true (FALSE if neither is true).
- **( ):** Parentheses group conditions to set all equal to TRUE.

```
SELECT *
FROM regions
WHERE (region = 'Americas'
    OR region = 'APAC')
    AND salesperson <> 'Anna Andreadi';
```

Solo Exercise:

Raw SQL

20 minutes

Follow the instructions in https://git.generalassemb.ly/ModernEngineering/raw-sql-superstore/blob/main/lab-1.md.

 © 2024 General Assembly

Solo Exercise:

Over to You | Solution

1) **How many office supplies products do we offer?**
   `SELECT * FROM products WHERE category = 'Office Supplies';` **5689**

2) **How many chairs do we offer?**
   `SELECT * FROM products WHERE sub_category = 'Chairs';` **619**

3) **What is the cheapest table we offer?**
   `SELECT * FROM products WHERE sub_category = 'Tables' ORDER BY 5 ASC;` **$12.18**

4) **How many office supplies products cost more than $500?**
   `SELECT * FROM products WHERE category = 'Office Supplies' AND product_cost_to_consumer > 500;` **58**

5) **How many product over $500 were technology or furniture?**
   `SELECT * FROM products WHERE (category = 'Technology' OR category = 'Furniture') AND product_cost_to_consumer > 500;` **114**

Comparison Operators

IN ()	Found in list of items.
BETWEEN	Within the range of, including boundaries.
NOT	Negates a condition.
LIKE, ILIKE	Contains item; ILIKE disregards case.
%	Wildcard, none to many characters.
_	Wildcard, single character.

 © 2024 General Assembly

Guided Walk-Through:
Filtering With IN

IN () allows you to specify multiple values in WHERE, while **NOT IN ()** allows you to negate the list of values.

1. Which products are **either** furnishings or technology (based on category name)?

```
SELECT * FROM products

WHERE category IN ('Furniture', 'Technology');
```

2. Which products are **neither** furnishings nor technology?

```
SELECT * FROM products

WHERE category NOT IN ('Furniture', 'Technology');
```

Guided Walk-Through:
Filtering With BETWEEN

BETWEEN () allows you to select values within a given range.

1. Which products have a cost to consumers between $25 and $100?

```
SELECT * FROM products
WHERE product_cost_to_consumer BETWEEN 25 AND 100;
```

2. Which orders have sales between $50 and $100?

```
SELECT * FROM orders
WHERE sales BETWEEN 50 AND 100;
```

 © 2024 General Assembly

Guided Walk-Through:
Filtering LIKE and ILIKE

LIKE is used for pattern matching in SQL, while NOT LIKE negates the match.

1. Which products have “Calculator” in the product name?

```sql
SELECT *
FROM products
WHERE product_name LIKE '%Calculator%';
```

2. Which products have “Printer” in the category name?

```sql
SELECT *
FROM products
WHERE product_name ILIKE '%PRINTER%';
```

Guided Walk-Through:
Filtering With Wildcards

PostgreSQL provides two wildcard characters to work with LIKE:

- **Percent (%)** for matching any sequence of characters.
- **Underscore (_)** for matching any single character.

1. Which products have “Clock” in the product name?

```
SELECT *
FROM products
WHERE product_name LIKE '%Clock%';
```

2. Which customers have names that start with “A” and third letter of “r”?

```
SELECT *
FROM customers
WHERE customer_name LIKE 'A_r%';
```

Querying Databases

From Stakeholder Questions to Efficient Queries

From Stakeholder Questions to SQL Queries

Example Stakeholder Question No. 1:

Who are the salespeople in the United States?

Resulting SQL query

```
SELECT DISTINCT salesperson
FROM regions
WHERE country ILIKE '%United States%';
```

 © 2024 General Assembly

From Stakeholder Questions to SQL Queries (Cont.)

Example Stakeholder Question No. 2:

What were the top five sales using second-class mail delivery?

Resulting SQL query

```
SELECT DISTINCT order_id, sales
FROM orders
WHERE ship_mode ILIKE '%second%'
ORDER BY 2 DESC
LIMIT 5;
```

From Stakeholder Questions to SQL Queries (Cont.)

Example Stakeholder Question No. 3:
Of orders with a discount between 25% and 50%, how many used standard shipping?

Resulting SQL query

```
SELECT *
FROM orders
WHERE discount BETWEEN .25 AND .5
AND ship_mode ILIKE '%standard%';
```

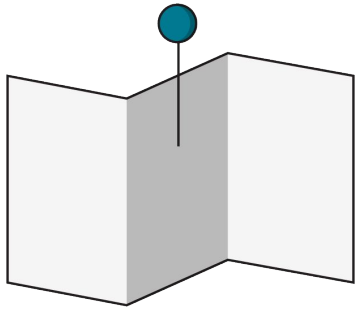
Partner Exercise:

From Questions to Queries

20 minutes

Together with your paired chat partner, write SQL queries for the following questions:

1. How many countries are in the “Americas” region?
2. Which tech products are sold to consumers in the $500–$1,000 range?
3. How many product items were returned in 2019 for unknown reasons?
   (Hint: Use ILIKE and the order_id field from the Returns table to solve for 2019.)



Partner Exercise:

From Questions to Queries | Solutions

1. How many countries are in the “Americas” region?

```
SELECT DISTINCT country FROM regions
WHERE region ILIKE '%America%';
```

2. Which tech products are sold to consumers in the $500–$1,000 range?

```
SELECT * FROM products WHERE category ILIKE '%tech%'
AND product_cost_to_consumer BETWEEN 500 AND 1000;
```

3. How many product items were returned in 2019 for unknown reasons (‘Not Given’)?

```
SELECT order_id FROM returns
WHERE order_id ILIKE '___2019%'
AND reason_returned = 'Not Given'
```

Solo Exercise:

Over to You

20 minutes

Follow the instructions in https://git.generalassemb.ly/ModernEngineering/raw-sql-superstore/blob/main/lab-2.md

Solo Exercise:

Over to You | Solution

1) **How many Canon copiers do we offer?**

```
SELECT * FROM products WHERE product_name ILIKE '%canon%';
```

113

2) **How many 3D printers do we offer?**

```
SELECT * FROM products WHERE product_name ILIKE '%3d%' AND product_name
ILIKE '%printer%';
```

4

3) **What is the most expensive Cisco phone?**

```
SELECT * FROM products WHERE product_name ILIKE '%cisco%' AND product_name
ILIKE '%phone%' ORDER BY 5 DESC;
```

$654.24

 © 2024 General Assembly

Solo Exercise:

Over to You | Solution (Cont.)

4) **How many products with “TEC” in the product_id cost between $400 and $600?**

```
SELECT * FROM products WHERE product_id ILIKE '%tec%' AND
product_cost_to_consumer BETWEEN 400 AND 600;
```

47

5) **How many products that cost between $500 and $1,000 are phones, machines, or copiers?**

```
SELECT * FROM products WHERE sub_category IN ('Copiers', 'Machines',
'Phones') AND product_cost_to_consumer BETWEEN 500 AND 1000;
```

61

 © 2024 General Assembly

Validating Your Output

How do you know your results are correct?

- Check that the table and column names are correct.
- Check that your single quotes (‘’) or double quotes (“”) are closed.
- Check operators such as >, <. =. !=, etc.
- Use EXPLAIN before SELECT to check how your SQL statement will access your database.
- Review error messages when your query does not run.
- Apply business context or industry knowledge (of similar data) to check order of magnitude.

 © 2024 General Assembly

Solo Exercise:

Extra Practice [Optional]

Follow the instructions in https://git.generalassemb.ly/ModernEngineering/raw-sql-superstore/blob/main/extra-practice.md .

 © 2024 General Assembly

Day 2

We Now Have...

- A set of requirements we can use to build our services
- API planning and understanding of the information we are manipulating with our modern application
- An environment to build in

 © 2024 General Assembly

Solo Exercise:

Daily Exit Ticket

3 minutes

- Please take a moment to give us your feedback after today's course!

- Use the QR code or follow the link: https://bit.ly/pruMEFc2

bitly

Scan Me!

 © 2024 General Assembly


GA